Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

Licence en Ingénierie de Développement d'Applications Informatiques 2023-2024

Jeu de carte marocaine

Réalisée par :

Yousra Ben Mokhtar.

Wissal Choukri.

Encadrée par:

P. IkramBENABDELOUAHAB

Remerciement

Nous souhaitons exprimer notre sincère gratitude, Prof. Ikram BENABDELOUAHAB, pour votre encadrement exceptionnel tout au long du module de langage de programmation C++. Votre enseignement a été une ressource inestimable, nous apportant un soutien précieux dans notre compréhension des concepts complexes et nous guidant habilement dans la résolution des problèmes. Nous tenons à vous remercier chaleureusement de nous avoir offert l'opportunité de réaliser ce projet. Nous sommes déterminés à mettre en pratique les connaissances acquises et à fournir un résultat de qualité. Votre manière remarquable d'expliquer et de diriger a grandement contribué à notre apprentissage, et nous sommes reconnaissants pour votre engagement envers notre réussite .

Table de matières

I. Introduction :

1- QT :

Qt est un framework de développement logiciel multiplateforme utilisé pour la création d'interfaces graphiques et d'applications logicielles. Développé par la société Qt Company, Qt offre une architecture robuste et des outils permettant aux programmeurs de concevoir des applications efficaces et esthétiques. Il est largement utilisé dans l'industrie du logiciel en raison de sa polyvalence, de sa facilité d'utilisation et de son support pour le développement d'applications multiplateformes, fonctionnant sur des systèmes d'exploitation tels que Windows, macOS, Linux et d'autres. Qt prend en charge le langage de programmation C++ et propose des fonctionnalités avancées telles que la gestion des ressources, la programmation multithread, et des widgets graphiques personnalisables.

a- Structure générale :

L'API Qt est constituée de classes aux noms préfixés par Q et dont chaque mot commence par une majuscule (ex: QLineEdit), c'est la typographie camel case. Ces classes ont souvent pour attributs des types énumérés déclarés dans l'espace de nommage Qt20. Mis à part une architecture en pur objet, certaines fonctionnalités basiques sont implémentées par des macros (chaîne de caractères à traduire avec tr, affichage sur la sortie standard avec qDebug…).

Les conventions de nommage des méthodes sont assez semblables à celles de Java : le lower camel case est utilisé, c'est-à-dire que tous les mots sauf le premier prennent une majuscule (ex: indicatorFollowsStyle()), les modificateurs sont précédés par set, en revanche les accesseurs prennent simplement le nom de l'attribut (ex : text()) ou commencent par is dans le cas des booléens (ex : isChecked()).

b- Arborescence des objets :

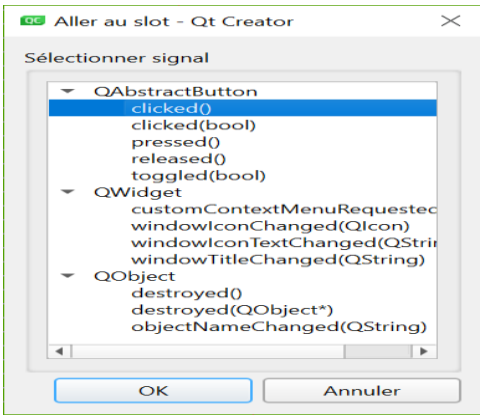
Les objets Qt (ceux héritant de QObject) peuvent s'organiser d'eux-mêmes sous forme d'arbre. Ainsi, lorsqu'une classe est instanciée, on peut lui définir un objet parent. Cette organisation des objets sous forme d'arbre facilite la gestion de la mémoire car avant qu'un objet parent ne soit détruit, Qt appelle récursivement le destructeur de tous les enfants 21.Cette notion d'arbre des objets permet également de déboguer plus facilement, via l'appel de méthodes comme QObject::dumpObjectTree() et Object::dumpObjectInfo()

c- Compilateur de méta-objets :

Le moc22 (pour Meta Object Compiler) est un préprocesseur qui, appliqué avant compilation du code source d'un programme Qt, génère des meta-informations relatives aux classes utilisées dans le programme. Ces meta-informations sont ensuite utilisées par Qt pour fournir des fonctions non disponibles en C++, comme les signaux et slots et l'introspection.

L'utilisation d'un tel outil additionnel démarque les programmes Qt du langage C++ standard. Ce fonctionnement est vu par Qt Développent Frameworks comme un compromis nécessaire pour fournir l'introspection et les mécanismes de signaux. À la sortie de Qt 1.x, les implémentations des templates par les compilateurs C++ n'étaient pas suffisamment homogènes.

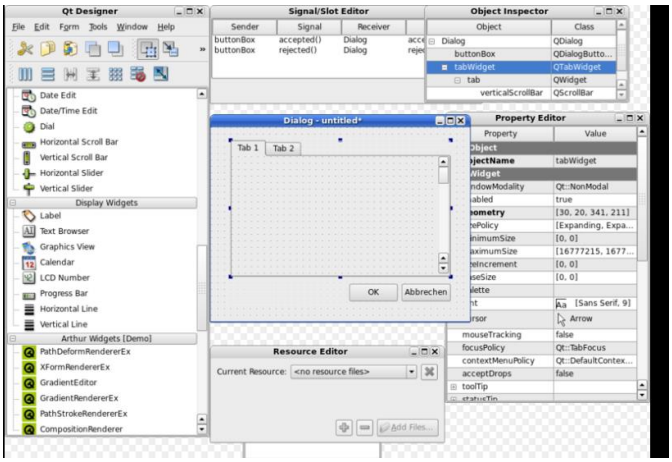
d-Signaux et slots :



Les signaux et slots sont une implémentation du patron de conception observateur. L'idée est de connecter des objets entre eux via des signaux qui sont émis et reçus par des slots. Du point de vue du développeur, les signaux sont représentés comme de simples méthodes de la classe émettrice, dont il n'y a pas d'implémentation. Ces « méthodes » sont par la suite appelées, en faisant précéder « emit », qui désigne l'émission du signal. Pour sa part, le slot connecté à un signal est une méthode de la classe réceptrice, qui doit avoir la même signature (autrement dit les mêmes paramètres que le signal auquel il est connecté), mais à la différence des signaux, il doit être implémenté par le développeur. Le code de cette implémentation représente les actions à réaliser à la réception du signal.

C'est le MOC qui se charge de générer le code C++ nécessaire pour connecter les signaux et les slots.

e- Concepteur interface :



Qt Designer est un logiciel qui permet de créer des interfaces graphiques Qt dans un environnement convivial. L'utilisateur, par glisser-déposer, place les composants d'interface graphique et y règle leurs propriétés facilement. Les fichiers d'interface graphique sont formatés en XML et portent l'extension .ui 24.

Lors de la compilation, un fichier d'interface graphique est converti en classe C++ par l'utilitaire uic. Il y a plusieurs manières pour le développeur d'employer cette classe 25 :

*l'instancier directement et connecter les signaux et slots .

*l'agréger au sein d'une autre classe

*l'hériter pour en faire une classe mère et ayant accès ainsi à tous les éléments constitutifs de l'interface créée

*la générer à la volée avec la classe QUiLoader qui se charge d'interpréter le fichier XML .ui et retourner une instance de classe QWidget

f-Qmake :

Qt se voulant un environnement de développement portable et ayant le MOC comme étape intermédiaire avant la phase de compilation/édition de liens, il a été nécessaire de concevoir un moteur de production spécifique. C'est ainsi qu'est conçu le programme qmake.

Ce dernier prend en entrée un fichier (avec l'extension .pro) décrivant le projet (liste des fichiers sources, dépendances, paramètres passés au compilateur, etc.) et génère un fichier de projet spécifique à la plateforme. Ainsi, sous les systèmes UNIX qmake produit un Makefile qui contient la liste des commandes à exécuter pour génération d'un exécutable, à l'exception des étapes spécifiques à Qt (génération des classes C++ lors de la conception d'interface graphique avec Qt Designer, génération du code C++ pour lier les signaux et les slots, ajout d'un fichier au projet, etc.).

Le fichier de projet est fait pour être très facilement éditable par un développeur. Il consiste en une série d'affectations de variables. En voici un exemple pour un petit projet:

```
TARGET = monAppli
SOURCES = main.cpp mainwindow.cpp
HEADERS = mainwindow.h
FORMS = mainwindow.ui
QT += sql
```

Ces déclarations demandent que l'exécutable soit nommé monAppli, donne la liste des fichiers sources, en-têtes et fichiers d'interface graphique. La dernière ligne déclare que le projet requiert le module SQL de Qt.

G- style :

La bibliothèque embarque divers thèmes de widgets qui lui donnent une bonne intégration visuelle sur toutes les plateformes. Sur les environnements de bureau GNOME, Mac OS X et Windows les applications Qt ont ainsi l'apparence d'applications natives.

Qt permet de personnaliser l'apparence des différents composants d'interface graphique en utilisant le principe des feuilles de style en cascade (CSS).

K-Outils de développement :

Qt Development Frameworks fournit un ensemble de logiciels libres pour faciliter le développement d'applications Qt :

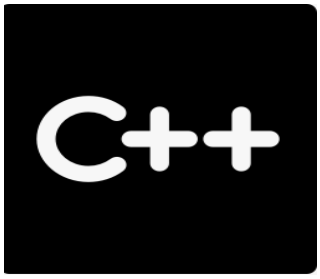
*Qt Designer est un concepteur d'interface graphique, il enregistre les fichiers .ui ;

*Qt Assistant permet de visualiser la documentation complète de Qt hors-ligne ;

*Qt Creator est l'environnement de développement intégré destiné à Qt et facilite la gestion d'un projet Qt. Son éditeur de texte offre les principales fonctions que sont la coloration syntaxique, le complètement, l'indentation, etc.

B – Langage c++ :

le langage C++ qui est un langage de programmation informatique créé par Bjarne Stroustrup, étendant le langage C avec des fonctionnalités de programmation orientée objet. Il offre un contrôle précis sur la gestion de la mémoire, ce qui permet une optimisation des performances. Sa polyvalence lui permet d'être utilisé dans divers domaines tels que les applications de bureau, les jeux vidéo et les logiciels embarqués. La bibliothèque standard (STL) fournit des outils prêts à l'emploi, et sa portabilité permet une exécution sur différentes plates-formes. En raison de sa puissance et de sa flexibilité, le C++ demeure un choix populaire dans l'industrie du développement logiciel.



II. Objectif de projet :

L'objectif fondamental de ce projet consiste à créer et mettre en œuvre un jeu de cartes marocain en utilisant la programmation orientée objet **(POO)** avec le **langage C++.** Pour enrichir l'expérience ludique, une interface graphique sera développée en utilisant le **Framework Qt**. Ce jeu devra permettre l'affrontement d'au moins deux joueurs, incluant un joueur humain (l'utilisateur) et un adversaire aléatoire (l'ordinateur).

III. Le mécanisme de jeu ;

Dans ce jeu de carte marocaine « *BENCHO* » une création singulière résultant de l'union des noms de réalisateurs de projet ,"BenMokhtar" et "Choukri".

le mécanisme fondamental repose sur la comparaison des valeurs des cartes entre le joueur et l'ordinateur : Chaque tour commence par le joueur initiant l'action en posant une carte de son jeu sur la table, exposant ainsi sa valeur respective. L'ordinateur réagit ensuite en dévoilant également une carte de sa main, révélant ainsi sa propre valeur. L'issue du tour est déterminée par la comparaison des valeurs des cartes. Si la carte de l'ordinateur a une valeur supérieure à celle du joueur, l'ordinateur remporte les deux cartes en jeu et les ajoute à son tas de points. Cependant, en cas d'égalité , l'ordinateur et le joueur gagnent un point pour chacun . L'accumulation de points est un aspect crucial du jeu, chaque joueur marquant des points en fonction des cartes qu'il remporte. Cette dynamique de pointage contribue à intensifier la compétition entre le joueur et l'ordinateur. La partie se poursuit ainsi, avec les joueurs évaluant stratégiquement la valeur de leurs cartes à chaque tour. La détermination du gagnant intervient à la fin de la partie, une fois que toutes les cartes ont été jouées. Le joueur ayant accumulé le plus grand nombre de cartes, et donc de points, est alors déclaré vainqueur. Ce mécanisme à trois niveaux de difficultés.

•Niveau de Difficulté Facile :

Au niveau facile, "BENCHO" offre une introduction accessible aux joueurs, facilitant ainsi la compréhension des règles et des mécaniques de base. Les valeurs numériques des cartes sont clairement définies, permettant aux participants de se familiariser progressivement avec la hiérarchie des cartes. Ce niveau met l'accent sur la convivialité, encourageant ainsi les joueurs novices à développer leurs compétences sans être accablés par des défis trop complexes.

•Niveau de Difficulté Moyen :

Au niveau moyen, "BENCHO" on introduit le jeu de manière plus stratégies. Le joueurs doivent maintenant considérer soigneusement les cartes qu'ils jouent, anticiper les mouvements de leurs adversaires et prendre des décisions tactiques pour maximiser leurs gains. La gestion habile des cartes devient cruciale, ajoutant une dimension intellectuelle et stimulante à l'expérience du jeu.

•Niveau de Difficulté Avancé :

Le niveau plus complexe de "BENCHO" les joueurs doivent avoir des méthodes plus intelligentes. Or Ce niveau exige une compréhension approfondie des stratégies avancées, une anticipation aiguisée des actions adverses, et une capacité à ajuster rapidement sa tactique en fonction du déroulement du jeu.

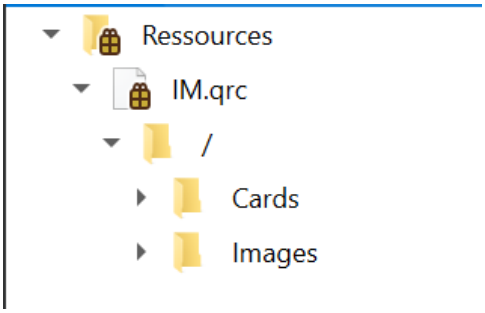
IV. *Fonctionnalités souhaitées :*

- Intégration d'un mode joueur contre ordinateur.
- Développement d'une interface graphique offrant une représentation visuelle récurrente du jeu, favorisant ainsi une compréhension continue pour l'utilisateur.
- Stockage de toutes les images de cartes du jeu du côté gauche de l'interface.
- Mise en place d'une fonctionnalité où le joueur, lorsqu'il clique sur un bouton dédié, reçoit 48 cartes pour débuter la partie.
- Lancement du jeu avec le joueur et l'ordinateur qui débutent leurs actions.
- Intégration de boutons de cartes, chacun illustrant visuellement une image de carte spécifique.
- Mise en place d'un logique du jeu .

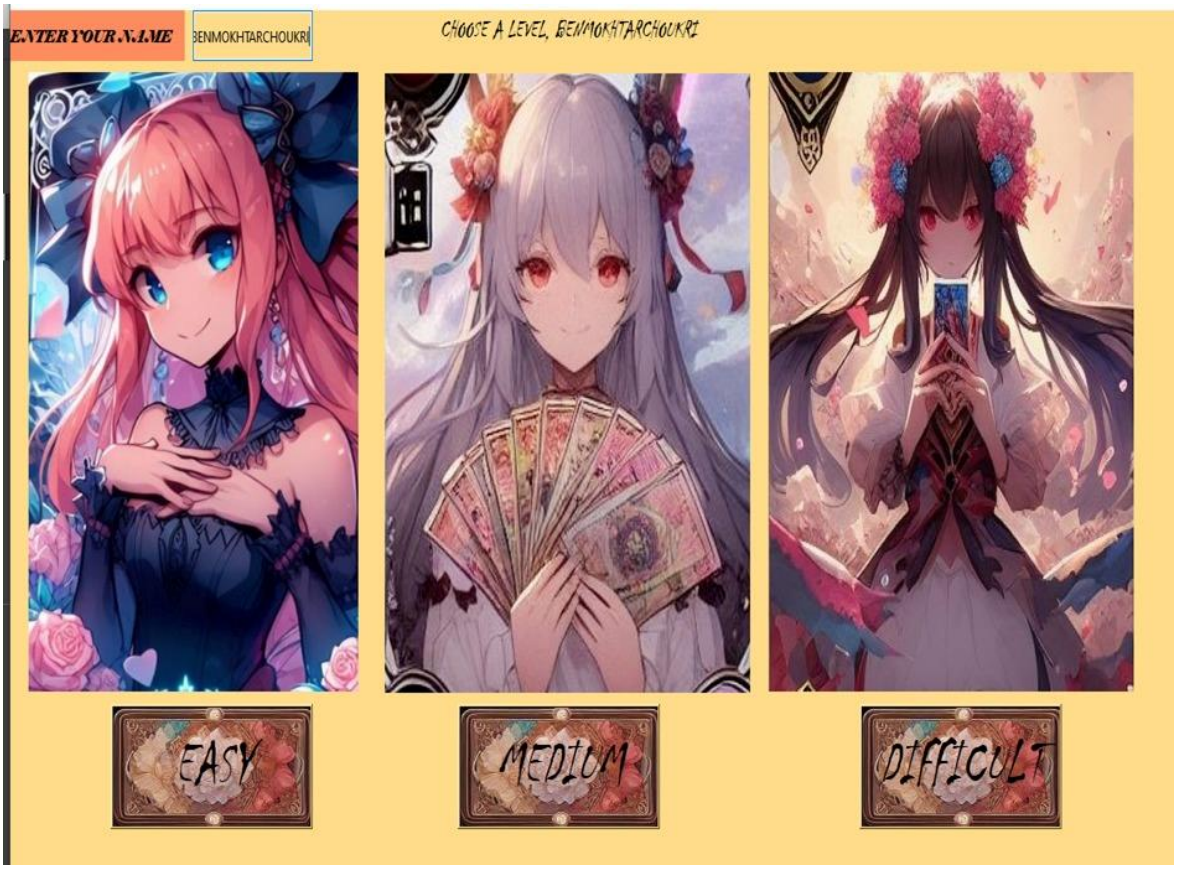
V. *Jeux(BENCHO):*

A. *Interface :*

La création de l'interface pour notre application Qt a débuté par une recherche minutieuse des cartes et des images à utiliser, provenant de diverses sources. Ces images ont été soigneusement organisées et stockées dans le dossier de ressources dédié de l'application, spécifiquement sous les sous-dossiers "cartes" et "images".



L'utilisation du fichier MainWindow.ui, généré par l'éditeur visuel Qt Designer, a été cruciale pour définir l'agencement de la fenêtre principale (MainWindow) de l'application Qt à travers un fichier XML. Dans cette fenêtre, une StackedWidget a été habilement intégrée, permettant l'insertion de plusieurs pages pour faciliter la gestion des différentes étapes du jeu.L'interface comprend également : une label pour afficher une image de fond, créant ainsi une atmosphère visuelle immersive, ainsi qu'une label dédiée au titre du jeu : Deux boutons ont été ajoutés pour la navigation, l'un pour passer à la page suivante nommée « START »et l'autre pour quitter la page en cours nommée « EXIT ». Lorsque le bouton « START » est activé, il redirige vers la page suivante qui représente les niveaux de difficulté :

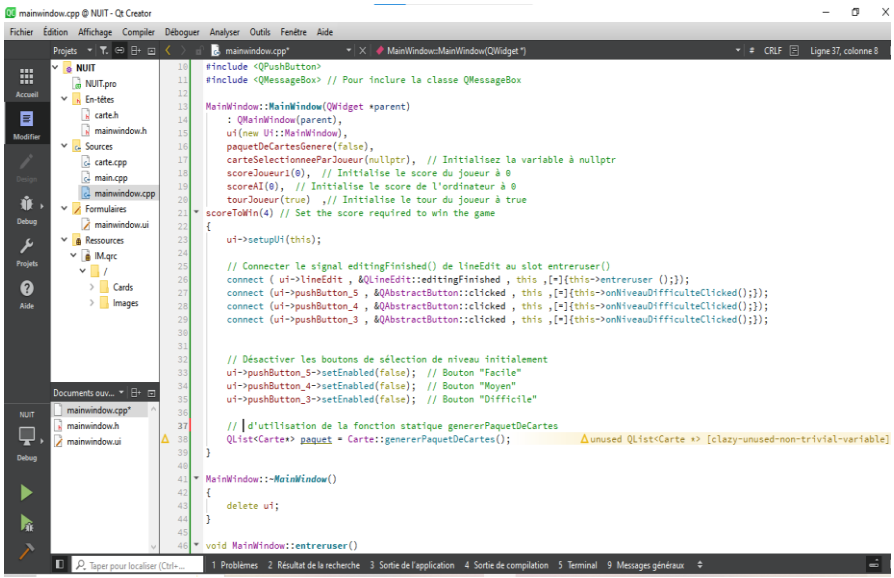


tels que "Easy", "Middle", et "Difficult". De plus, un LineEdit et une Label ont été intégrés sur cette page, incitant l'utilisateur à saisir son nom avant de jouer. La fonction "entrerUser()" a été implémentée pour activer les boutons une fois que l'utilisateur a entré son nom, Un message incitant le choix du niveau s'affiche, et en cliquant sur le niveau souhaité, par exemple, "Easy", l'utilisateur est dirigé vers une nouvelle page contenant le jeu avec distribution des cartes et l'initiation du jeu. Voici les capture d'écrans sous-dessus pour bien comprendre .

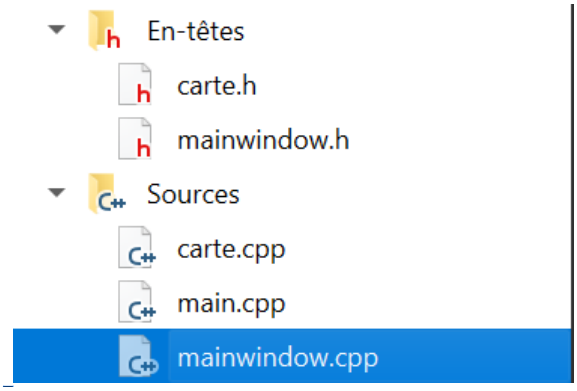
B. Distribution des cartes :

➢ Interface :

➢ Code :



Pour realiser la distribution des cartes on a crée premièrement fichier cpp qui comporte une classe <carte> qui regroupe les 48 cartes dans une listes et son fichier header :



Carte.cpp :

```cpp
#include "carte.h"

Carte::Carte(QString type, int valeur, QString chemin, int indice)
    : type(type), valeur(valeur), chemin(chemin), indice(indice)
{
}

QString Carte::getType() const
{
    return type;
}

int Carte::getValeur() const
{
    return valeur;
}

QString Carte::getChemin() const
{
    return chemin;
}
int Carte::getIndice() const
{
    return indice;
}


QList<Carte*> Carte::genererPaquetDeCartes()
{
    QList<Carte*> paquetDeCartes;

    // Cartes de type B
    for (int i = 1; i <= 12; ++i) {
        QString chemin = QString(":/Cards/B.%1.jpg").arg(i);
        paquetDeCartes << new Carte("B", i, chemin, i );
    }

    // Cartes de type C
    for (int i = 1; i <= 12; ++i) {
        QString chemin = QString(":/Cards/C.%1.jpg").arg(i);
        paquetDeCartes << new Carte("C", i, chemin, i + 12);
    }

    // Cartes de type O
    for (int i = 1; i <= 12; ++i) {
        QString chemin = QString(":/Cards/O.%1.jpg").arg(i);
        paquetDeCartes << new Carte("O", i, chemin, i + 24);
    }

```

```
    // Cartes de type P
    for (int i = 1; i <= 12; ++i) {
        QString chemin = QString(":/Cards/P.%1.jpg").arg(i);
        paquetDeCartes << new Carte("P", i, chemin, i + 36);
    }

    return paquetDeCartes;
}
```

Carte.h :

```
#ifndef CARD_H
#define CARD_H

#include <QString>
#include <QList>

class Carte
{
public:
    Carte(QString type, int valeur, QString chemin,int indice);
    QString getType() const;
    int getValeur() const;
    QString getChemin() const;
    int getIndice() const;
    static QList<Carte*> genererPaquetDeCartes();

private:
    QString type;
    int valeur;
    QString chemin;
    int indice;
};

#endif // CARD_H
```

La methode distribuerCartes() :

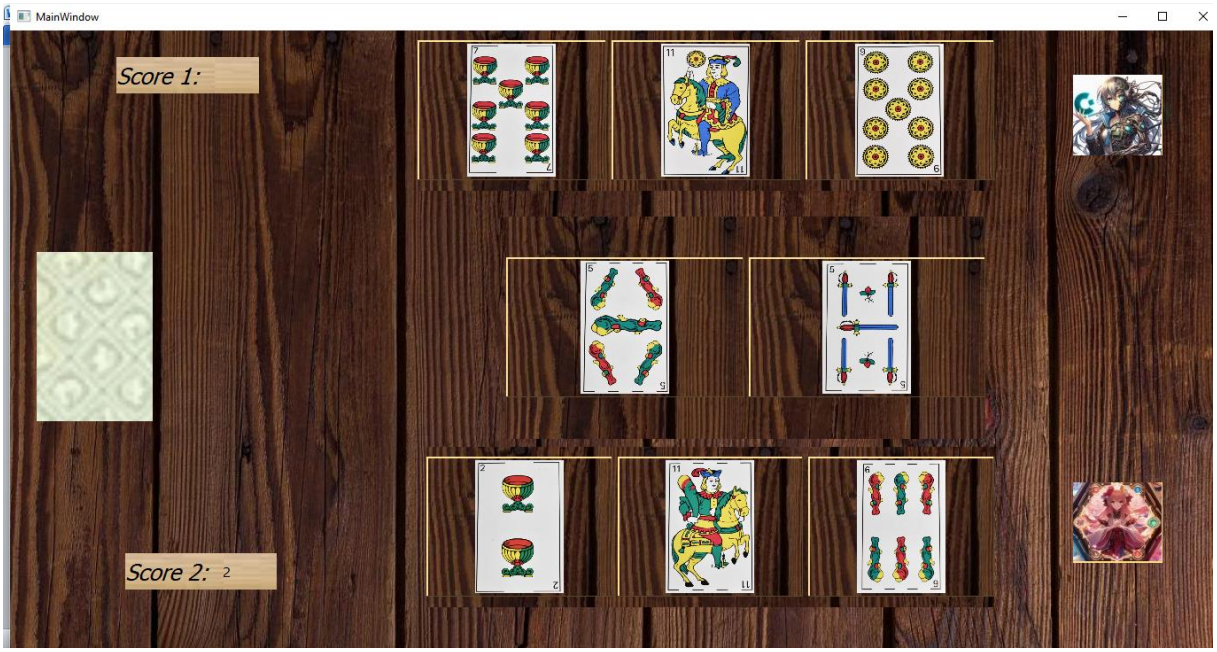
```
void MainWindow::distribuerCartes()
{
    for (int i = 0; i < 8; ++i) {

        QString nomBouton = "pushButton_" + QString::number(i + 6);
        QPushButton *bouton = findChild<QPushButton *>(nomBouton);
        QPixmap cartePixmap(paquetDeCartes[i]->getChemin());

        // Redimensionnez l'image
        cartePixmap = cartePixmap.scaled(QSize(100, 150));

        bouton->setIcon(QIcon(cartePixmap));
        bouton->setIconSize(cartePixmap.size());

        // Ajoutez les 4 premiers boutons à layoutCartes1
        if (i < 4) {
            ui->layoutCartes1->addWidget(bouton);
        }
        // Ajoutez les 4 boutons suivants à layoutCartes2
        else {
            ui->layoutCartes2->addWidget(bouton);
        }

        // Connectez le clic de l'utilisateur sur le bouton
        connect(bouton, &QPushButton::clicked, this, [=](){
            onCarteButtonClicked(bouton);
        });
    }
}
```

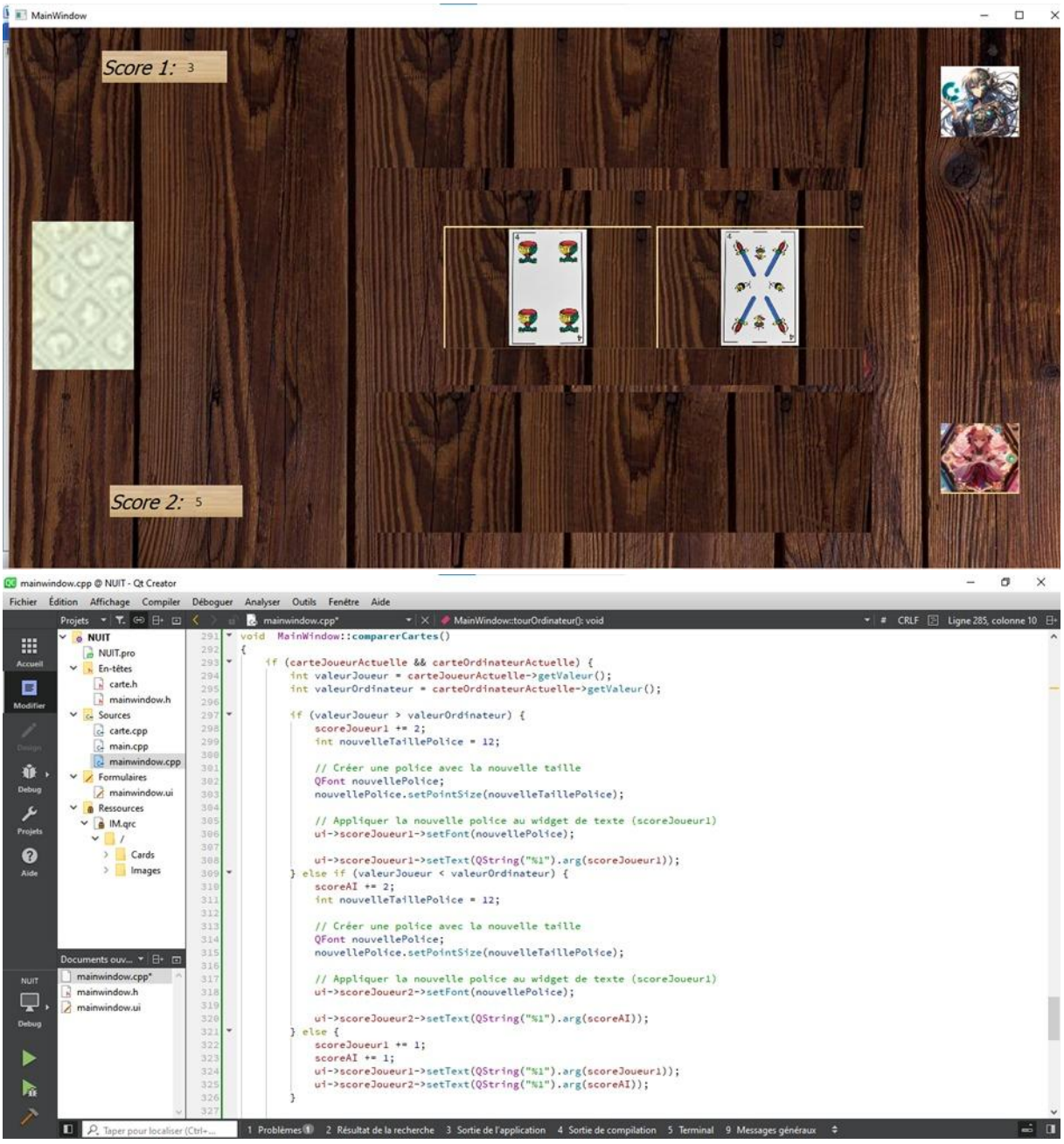
Apres on va appeler les fonction genererPaquetDeCartes() et distribuerCartes() dans mainwindow.cpp dans la methode onNiveauDifficulteClicked() :

```
void MainWindow::onNiveauDifficulteClicked()
{
    // Vérifiez si le paquet a déjà été généré
    if (!paquetDeCartesGenere) {
        // Appelez votre fonction genererPaquetDeCartes ici
        paquetDeCartes = Carte::genererPaquetDeCartes();

        qDebug() << "Nombre de cartes générées : " << paquetDeCartes.size();

        // Mélangez aléatoirement le paquet
        std::random_shuffle(paquetDeCartes.begin(), paquetDeCartes.end());

        // Chargez l'image de la carte "back"
        QPixmap backCardPixmap(":/Cards/back.jpg");

        // Affichez l'image de la carte "back" en tant qu'icône du bouton
        ui->pushButtonGenererPaquet->setIcon(QIcon(backCardPixmap));
        ui->pushButtonGenererPaquet->setIconSize(backCardPixmap.size());

        // Mettez à jour l'indicateur pour indiquer que le paquet a été généré
        paquetDeCartesGenere = true;
        // Distribuez les cartes sur les layouts
        distribuerCartes();
    }

}
```

Pour que chaque fois on click sur l'un des niveau de difficultés la distribution de cartes fonctionne directement .

C .Affichage de points :






```cpp
void MainWindow::comparerCartes()
{
    if (carteJoueurActuelle && carteOrdinateurActuelle) {
        int valeurJoueur = carteJoueurActuelle->getValeur();
        int valeurOrdinateur = carteOrdinateurActuelle->getValeur();

        if (valeurJoueur > valeurOrdinateur) {
            scoreJoueur1 += 2;
            int nouvelleTaillePolice = 12;

            // Créer une police avec la nouvelle taille
            QFont nouvellePolice;
            nouvellePolice.setPointSize(nouvelleTaillePolice);

            // Appliquer la nouvelle police au widget de texte (scoreJoueur1)
            ui->scoreJoueur1->setFont(nouvellePolice);

            ui->scoreJoueur1->setText(QString("%1").arg(scoreJoueur1));
        } else if (valeurJoueur < valeurOrdinateur) {
            scoreAI += 2;
            int nouvelleTaillePolice = 12;

            // Créer une police avec la nouvelle taille
            QFont nouvellePolice;
            nouvellePolice.setPointSize(nouvelleTaillePolice);

            // Appliquer la nouvelle police au widget de texte (scoreJoueur1)
            ui->scoreJoueur2->setFont(nouvellePolice);

            ui->scoreJoueur2->setText(QString("%1").arg(scoreAI));
        } else {
            scoreJoueur1 += 1;
            scoreAI += 1;
            ui->scoreJoueur1->setText(QString("%1").arg(scoreJoueur1));
            ui->scoreJoueur2->setText(QString("%1").arg(scoreAI));
        }
```

Pour La fonction comparerCartes() est appelée lorsque le joueur et l'ordinateur ont sélectionné une carte chacun. Elle compare les valeurs des cartes, met à jour les scores en conséquence, ajuste la police du texte affichant les scores, réinitialise les cartes actuelles, puis relance le tour du joueur après une pause d'une seconde.

La condition initiale vérifie si les cartes actuelles du joueur et de l'ordinateur existent. Ensuite, les valeurs des cartes sont extraites, et en fonction de leur comparaison, les scores des joueurs sont ajustés. La taille de la police des scores est également modifiée, et les widgets de texte affichant les scores sont mis à jour.

Les cartes actuelles sont ensuite réinitialisées, et le tour du joueur est relancé après une pause d'une seconde, permettant d'ajouter des logiques supplémentaires pour le tour du joueur si nécessaire. En résumé, cette fonction gère la comparaison des cartes, la mise à jour des scores, l'ajustement de la présentation visuelle, et prépare le terrain pour le prochain tour du joueur dans un jeu de cartes.

VI. Conclusion :

En résumé, "BENCHO", le jeu de cartes marocain, offre une expérience compétitive où la comparaison des valeurs des cartes entre le joueur et l'ordinateur crée une dynamique captivante. Les niveaux de difficulté progressifs permettent aux joueurs de développer leurs compétences stratégiques, allant de l'apprentissage des bases à la maîtrise de tactiques avancées. La détermination du vainqueur repose sur la capacité à accumuler des points grâce à des choix stratégiques judicieux. Dans l'ensemble, "BENCHO" se distingue comme un jeu engageant, offrant une combinaison unique de tradition culturelle et de défis intellectuels.

VII. Bibliographie :

https://www.bing.com/images/create?FORM=GENILP

https://www.canva.com/fr_fr/modeles/?query=INTERFACE-DES-JEUX-(facile-%2Cmoyen-%2Cdifficile-)

https://fr.freepik.com/search?format=search&query=jeux%20des%20cartes%20marocaines%20facile%20moyen%20difficile%20animm%C3%A9&type=photo

https://fr.wikipedia.org/wiki/Qt#Arborescence_des_objets

https://chat.openai.com/

https://www.flaticon.com/free-icon/c-logo_74897

https://www.youtube.com/watch?v=E4KUri4UdGM&list=PL6ujmQXEKRmjb7qB4kZZTYdSmfd0MGJf9

https://www.youtube.com/watch?v=iowPI7VpWLA&list=PL8ThI0DA8FbUBpEihprYsoQJSbcLm40nZ

https://www.shutterstock.com/fr/catalog

https://new.express.adobe.com/id/urn:aaid:sc:EU:4bc49e58-0145-47aa-869c-7e7dd3014a7e?category=search

https://www.bing.com/images/create?FORM=GENILP

https://www.youtube.com/watch?v=6IY3Ei-_n0o

https://perso.telecom-paristech.fr/elc/qt/Qt-Graphique.pdf

https://www.bing.com/images/create?FORM=GENILP

https://www.canva.com/fr_fr/modeles/?query=INTERFACE-DES-JEUX-(facile-%2Cmoyen-%2Cdifficile-)

https://fr.freepik.com/search?format=search&query=jeux%20des%20cartes%20marocaines%20facile%20moyen%20difficile%20animm%C3%A9&type=photo